TB3191

I²C Master Mode

Introduction

Author: Christopher Best, Microchip Technology Inc.

Inter-Integrated Circuit, more commonly referred to as I²C, is a synchronous, two-wire, bidirectional serial communications bus. The I²C module can be used to communicate with other I²C compatible EEPROMs, display drivers, sensors, or other microcontroller devices. This technical brief will discuss the features and functions of the stand-alone I²C module in Master mode. The stand-alone I²C module will not be confused with the traditional Master Synchronous Serial Port (MSSP), which contained both I²C and Serial Peripheral Interface (SPI) functions.

I²C Module Modes and Features

The I²C module provides the following operational modes and features:

- Master Mode
- Slave Mode with Byte NACKing
- Multi-Master Mode
- Dedicated Receive and Transmit Buffers
- Up to Four Dedicated Slave Address Buffers(1)
- 7-Bit and 10-Bit Addressing with Masking
- General Call Addressing
- Interrupts
- Bus Collision Detection
- Bus Time-Out Detection with Programmable Sources
- SDA Hold Time Selection
- Programmable Bus-Free Time Selection
- I²C, SMBus 2.0, and SMBus 3.0 Input Level Selection
- Direct Memory Access (DMA) Support(2)

Note:

1. Support for four dedicated slave buffers is only available when in 7-bit Addressing mode. When in 10-bit Addressing mode, only two dedicated address buffers are available.
2. Direct Memory Access (DMA) is not available on all devices. Please refer to the device data sheet to determine if the DMA is available.

Table of Contents

1. I²C Specification

The I²C specification was developed by Phillips Semiconductors to communicate between devices connected to a two-wire bus. Phillips recognized that there were many similarities between consumer electronics, industrial electronics, and telecommunications designs. Since the various designs often contained similar components, such as Analog-to-Digital converters (ADCs), LCDs, or EEPROMs, Phillips determined that they could simplify system design and maximize hardware efficiency by creating a communication bus that could be used to transfer data between any device connected to the bus.

This allowed designers to use devices from multiple manufacturers, or use one device in several designs. The specification also solved interfacing problems by creating a scheme that is now held as an industry standard, meaning any I²C device could communicate with any other I²C device without having to change the hardware or firmware of either device.

The I²C specification defines the bus as a two-wire, bidirectional communications scheme. One line carries the serial data (SDA), and one line carries the serial clock (SCL). Each I²C device has its unique address, either 7-bits or 10-bits in length. An I²C device can operate as either a bus master, bus slave, or both, depending on the device and application. The specification defines the data transfer rates as follows:

- Standard mode – transfer rates up to 100 Kbits/s
- Fast mode – transfer rates up to 400 Kbits/s
- Fast mode Plus – transfer rates up to 1 Mbit/s
- High-speed mode – transfer rates up to 3.4 Mbits/s

Microchip's I²C module implements master and slave hardware that supports Standard mode, Fast mode and Fast mode Plus. Throughout this technical brief, the I²C specification will be referred to so that the reader understands both the I²C module and the I²C specification.

I²C Bus Terminology

To properly understand the language used in the specification, the following is a list of terms commonly used by the specification and found throughout this technical brief:

Table 1-1. I²C Bus Terminology

Term	Description
Transmitter	The device that shifts data out onto the bus
Receiver	The device that shifts data in from the bus
Master	The device that initiates data transfer, generates the clock signal, and terminates transmission
Slave	The device addressed by the master
Multi-master	A bus with more than one device that can initiate data transfers
Arbitration	Procedure that ensures that only one master at a time controls the bus
Synchronization	Procedure to synchronize the clocks of two or more devices on the bus
Idle	Both the SDA and SCK lines are in a logic High state; no activity on the bus
Active	Any time in which one or more master devices are controlling the bus

.....continued

Term	Description
Address Slave	Slave device that has received a matching address and is actively being clocked by a master
Matching Address	Address byte clocked into a slave that matches the value stored in one of the I2CxADR registers
Write Request	Master sends an address byte with the R/$\overline{W}$ bit clear; master intends to write data to the slave
Read Request	Master sends an address byte with the R/$\overline{W}$ bit set; master intends to receive data from a slave
Clock Stretching	When a device holds the clock line low to pause communications
Bus Collision	Condition in which the expected data on SDA is a logic high, but is sampled as a logic low
Bus Time-Out	Condition in which a device on the bus is holding the bus for longer than a specified period

2. I²C Module Overview

The I²C module provides a synchronous serial interface between the microcontroller and other I²C compatible devices using the two-wire bus. The two signal connections, Serial Clock (SCL) and Serial Data (SDA), are bidirectional open-drain lines, each requiring pull-up resistors to the supply voltage. Pulling the line to ground is considered a logic '0', while allowing the line to float is considered a logic '1'.

Important: Note that the voltage levels of the logic '0' (low) and logic '1' (high) are not fixed and are dependent on the bus supply voltage.

According to the I²C specification, a logic input low level is up to 30% of V_{DD} ($V_{IL} \leq 0.3\ V_{DD}$), while a logic input high level is 70% to 100% of V_{DD} ($V_{IH} \geq 0.7\ V_{DD}$). Some legacy devices may use the previously defined fixed levels of V_{IL} = 1.5V and V_{IH} = 3.0V, but all new I²C compatible devices require the use of the 30/70% specification.

All I²C communication is performed using an 8-bit data word and a 1-bit Acknowledge sequence. All transactions on the bus are initiated and terminated by the master device. Depending on the direction of the data being transferred, there are four main operations performed by the I²C module:

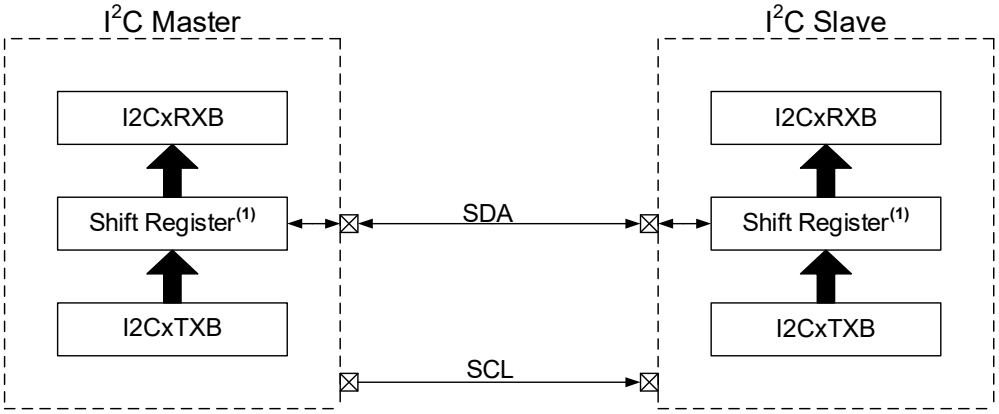
- Master Transmit – master is transmitting data to a slave
- Master Receive – master is receiving data from a slave
- Slave Transmit – slave is transmitting data to a master
- Slave Receive – slave is receiving data from a master

The I²C interface allows for a multi-master bus, meaning that there can be several master devices present on one bus. A master can select a slave device by transmitting an unique address on the bus. When the address matches a slave's address, the slave responds with an Acknowledge sequence ($\overline{ACK}$), and communication between the master and that slave can commence. All other devices connected to the bus must ignore any transactions not intended for them.

2.1 Dedicated Transmit/Receive Buffers

The I²C module has two dedicated data buffers, one for transmission (I2CxTXB) and one for reception (I2CxRXB) - see figure below.

Figure 2-1. I²C Transmit (I2CxTXB) and Receive (I2CxRXB) Buffers



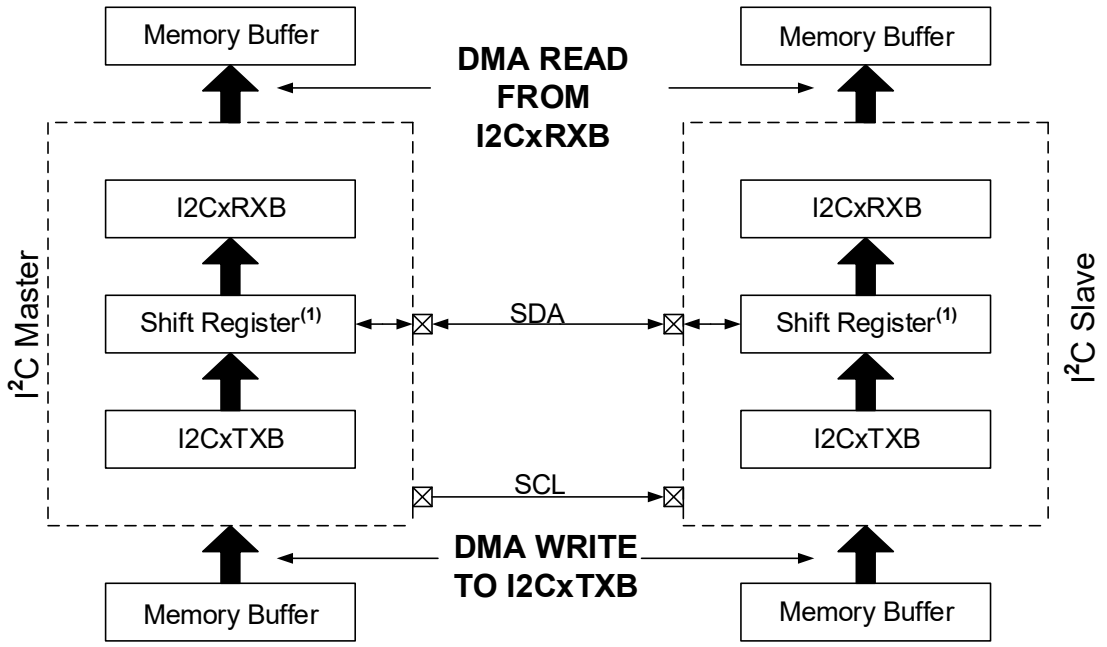
Note:

1. Shift register is not accessible to the user.

The transmit buffer, I2CxTXB, is loaded from software or from the Direct Memory Access (DMA) module (see Figure 2-2). When transmission begins, data loaded into the I2CxTXB is shifted into the transmit shift register and out onto the bus. The I2CxTXB can be loaded when the Transmit Buffer Empty Status (TXBE) bit of the I2CxSTAT1 register is set (TXBE = 1), indicating that the buffer is empty. When the buffer is empty and the I2CxCNT register is not equal to zero (I2CxCNT != 0), the I2C Transmit Interrupt Flag (I2CxTXIF) bit is set (I2CxTXIF = 0), and the generic I2C Interrupt Flag (I2CxIF) bit is also set if the I2C Transmit Interrupt Enable (I2CxTXIE) bit is set (I2CxTXIE = 1). Loading a new byte of data into the I2CxTXB clears the I2CxTXIF Flag bit. If the buffer is loaded when it is full (TXBE = 0), the Transmit Write Error Status (TXWE) bit is set, and the new data is discarded. If TXWE is set, user software must clear this Error condition to resume normal operation.

The receive buffer, I2CxRXB, receives data from the bus via the receive shift register. I2CxRXB can be read through user software or through the DMA (see Figure 2-2). When a new byte is received into I2CxRXB, the Receive Buffer Full Status (RXBF) bit of the I2CxSTAT1 register and the I2C Receive Interrupt Flag (I2CxRXIF) bit are set, and the generic I2CxIF is also set if the I2C Receive Interrupt Enable (I2CxRXIE) is set. Reading the buffer clears both RXBF and I2CxRXIF. If the buffer is read when it is empty (RXBF = 0), the Receive Read Error Status (RXRE) bit is set, and a Not Acknowledge (NACK) is generated. User software must clear the Error condition to resume normal operation.

Figure 2-2. I²C Transmit/Receive Buffers with DMA



Note:

1. Shift register is not accessible to the user.

Both transmit and receive buffers can be cleared by setting the Clear Buffer (CLRBF) bit of the I2CxSTAT1 register, which also clears both the I2CxTXIF and I2CxRXIF Interrupt flags.

2.2 Address Buffers

The I²C module has two address buffer registers, I2CxADB0 and I2CxADB1, which can be used as receive buffers in Slave mode, transmit buffers in Master mode, or both transmit and receive buffers in Multi-Master mode (see table below). This differs from the MSSP module in that the MSSP module only used the SSPBUF to receive or transmit an address (or data). The address buffers are enabled via the Address Buffer Disable (ABD) bit. When ABD is clear (ABD = 0), the address buffers are enabled; when the ABD is set (ABD = 1), the address buffers are disabled.

Table 2-1. *Address Buffer Direction for Master Modes*

Modes	MODE<2:0>	I2CxADB0	I2CxADB1
Master (7-bit)	100	Unused	TX
Master (10-bit)	101	TX	TX
Multi-Master (7-bit)	111	RX	TX

In 7-bit Master mode, I2CxADB1 is used to store a slave address, while I2CxADB0 is unused. When the address buffers are enabled (the ABD bit of I2CxCON2 = 0), the address loaded into I2CxADB1 is copied

into the transmit shift register automatically by hardware. Conversely, when the address buffers are disabled (ABD = 1), neither I2CxADB0 or I2CxADB1 are used, and the slave address is loaded into the I2CxTXB register by user software.

In 10-bit Master mode, I2CxADB0 is used to store the lower eight bits of the slave address, while I2CxADB1 is used to store the upper bits and R/$\overline{W}$ value of the slave address. When the address buffers are enabled (ABD = 0), the upper byte of the 10-bit address loaded into I2CxADB1 is copied automatically by the hardware into the transmit shift register. Once the master receives the $\overline{ACK}$ from the slave, the lower byte of the 10-bit address loaded into I2CxADB0 is copied automatically by the hardware into the transmit shift register.

In Multi-Master mode, only 7-bit addresses are used. If the device is addressed as a slave, the received matching slave address is copied into the I2CxADB0 register. If the device is communicating as a master, the contents of the I2CxADB1 register are copied into the transmit shift register to address the slave.

2.3 Receive Buffer

The stand-alone I^{2}C module has a dedicated receive buffer, I2CxRXB, which operates independently from the transmit buffer.

The receive buffer holds one byte of data that is shifted in from the receive shift register. User software or the DMA can read the byte through the I2CxRXB register. When a new byte is received, the Receive Buffer Full Status (RXBF) bit is set. The RXBF bit replaces the Buffer Full (BF) bit used in the MSSP module upon reception of a full byte. Reading I2CxRXB will clear the RXBF bit. If the buffer is read while empty (RXBF = 0), the Receive Read Error Status (RXRE) bit is set, and the module generates a NACK. User software must clear the RXRE bit to resume normal operation. Additionally, setting the Clear Buffer (CLRBF) bit clears both the receive and transmit buffers, as well as the I2C Receive Interrupt Flag (I2CxRXIF) bit and I2C Transmit Interrupt Flag (I2CxTXIF) bit.

2.4 Transmit Buffer

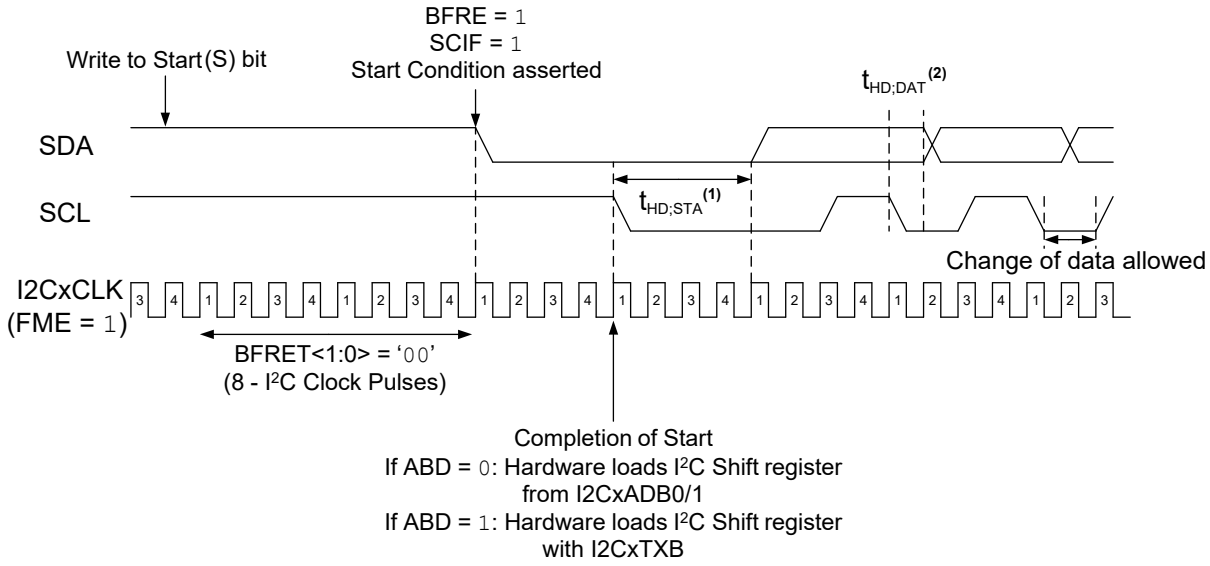
The stand-alone I^{2}C module has a dedicated transmit buffer, I2CxTXB, which operates independently from the receive buffer.

The transmit buffer is loaded with an address or data byte that is to be shifted into the transmit shift register and transmitted onto the bus. When the I2CxTXB is empty, the Transmit Buffer Empty Status (TXBE) bit is set, allowing user software or the DMA to load another byte into the buffer. Once the data is transmitted from the I2CxTXB register, the TXBE bit is cleared. If user software attempts to load the I2CxTXB while it is full (TXBF = 1), the Transmit Write Error Status (TXWE) bit is set, a NACK is generated, and the new data is ignored. If the TXWE Flag is set, software must clear this bit before attempting to load the buffer again. Additionally, setting the Clear Buffer (CLRBF) bit clears both the transmit and receive buffers, as well as the I2C Transmit Interrupt Flag (I2CxTXIF) bit and I2C Receive Interrupt Flag (I2CxRXIF) bit.

2.5 Start Condition

The I^{2}C specification defines a Start condition as the transition of the SDA line from an idle state (logic high level) to an active state (logic low level) while the SCL line is idle (see figure below). The Start condition is always initiated by the master and signifies the beginning of a transmission.

Figure 2-3. Start Condition



Note:

1. See device data sheet for Start condition hold time parameters.
2. SDA hold time are configured via the SDAHT<1:0> bits.

According to the I²C specification, a bus collision cannot occur on a Start condition. The Bus Free (BFRE) bit is used by module hardware to indicate the status of the bus. The Bus Free Time (BFRET<1:0>) bits define the amount of I²C clock cycles that master hardware must detect while the bus is idle before the BFRE bit is asserted. When the BFRE bit is set (BFRE = 1), the bus is considered in an idle state, and a master device may issue a Start condition. If there is more than one master on the bus (Multi-Master mode), and both attempt to issue a Start condition simultaneously, a bus collision will occur during the addressing phase of communication.

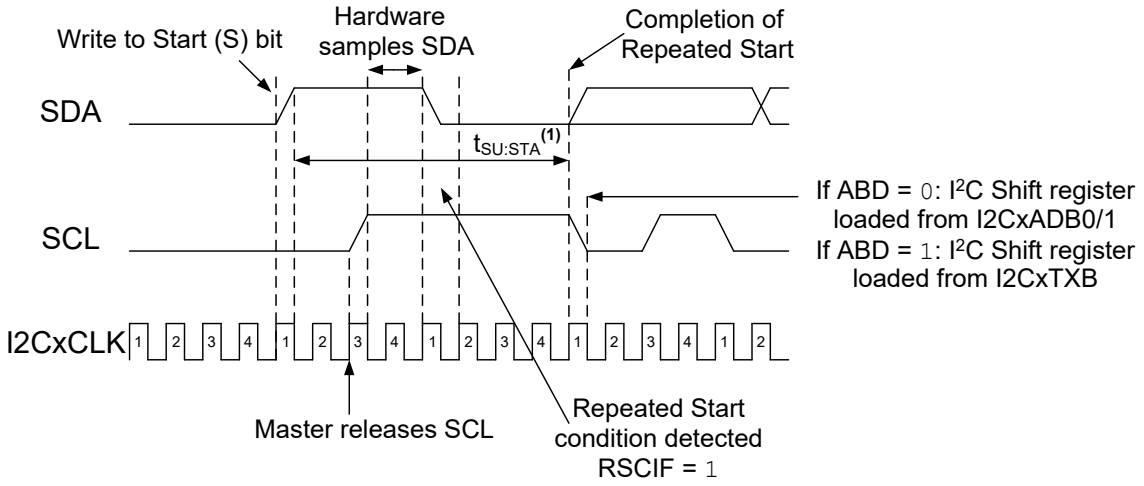
2.6 Repeated Start/Restart Condition

A Repeated Start or Restart condition is identical to a Start condition. A master device can issue a Restart condition instead of a Stop condition if it intends to hold the bus after completing the current data transfer. A Restart condition has the same effect on the slave as a Start condition would, resetting all slave logic and preparing it to receive an address. The Restart condition is always initiated by the master.

A Restart condition occurs when the Restart Enable (RSEN) bit is set, I2CxCNT is '0', and either master hardware or user software sets the Start bit.

When the Start bit is set, master hardware releases SDA (SDA floats high) for $T_{SCL}/2$. Then, hardware releases SCL for $T_{SCL}/2$, and samples SDA. If SDA is sampled low (while SCL is high), as bus collision has occurred, setting the Bus Collision Detect Interrupt Flag (BCLIF) bit and placing master hardware in the idle state. If SDA is sampled high (while SCL is also high), master hardware issues a Start condition. If ABD = 0, hardware loads the I²C shift register with the address loaded into I2CxADB0/1. If ABD = 1, hardware transfers the address from I2CxTXB into the shift register. Once a Restart condition is detected on the bus, the Restart Condition Interrupt Flag (RSCIF) bit is set. See figure below for more details.

Figure 2-4. Restart Condition



Note:

1. See device data sheet for Restart condition setup times.

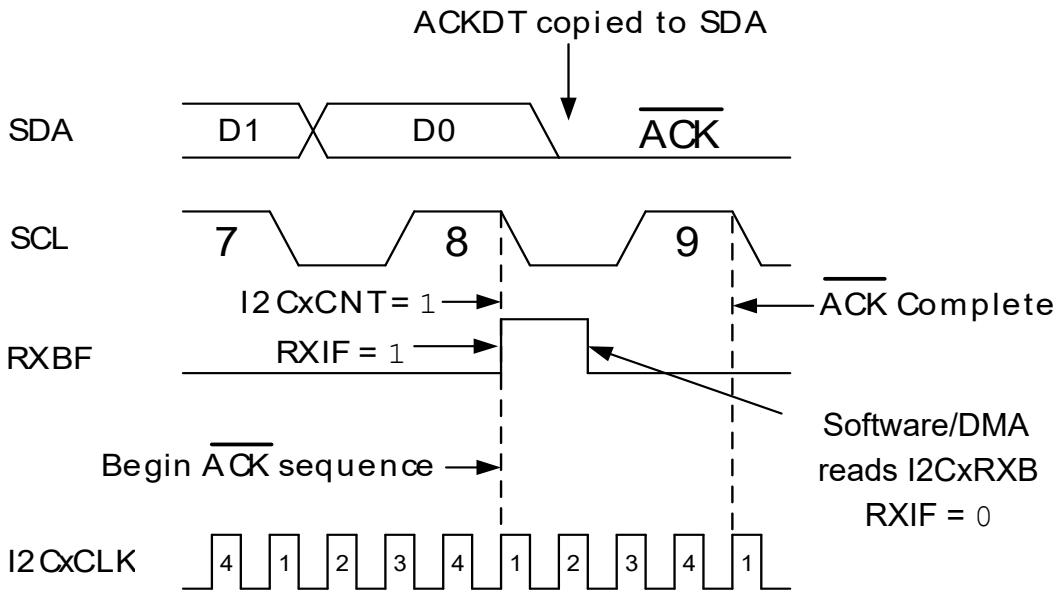
2.7 Acknowledge ($\overline{ACK}$)/Not Acknowledge (NACK) Sequence

The I^{2}C specification defines the Acknowledge sequence as a logic low state of the SDA line during the 9th SCL pulse for any successfully transferred byte. During this time, the transmitter must relinquish control of the SDA line to the receiver. The receiver must then pull the SDA line low and keep it low during the high period of the 9th SCL pulse.

When the receiver has successfully received a matching address byte or a valid data byte, it will pull the SDA line low during the 9th SCL pulse, which indicates to the transmitter that is has successfully received the information and is ready for the next byte.

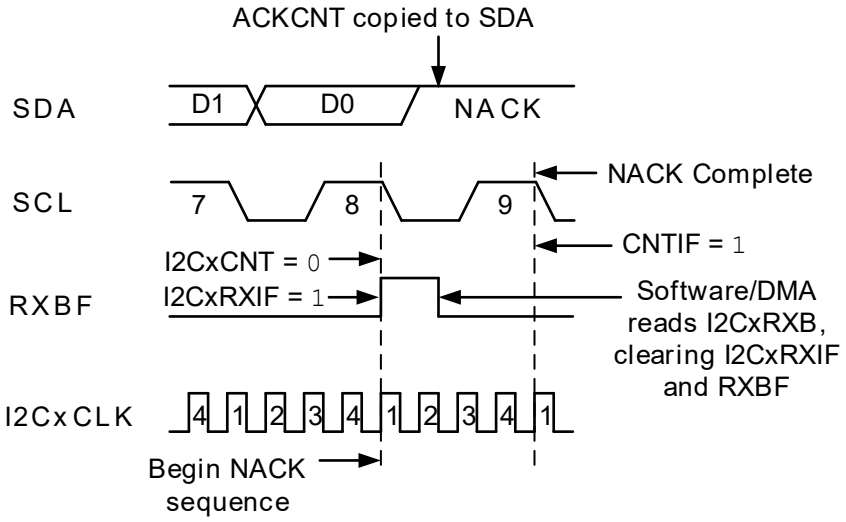
An Acknowledge sequence is enabled automatically by hardware following an address/data byte reception. On the 8th falling edge of SCL, the contents of either the Acknowledge Data (ACKDT) bit or the Acknowledge End of Count (ACKCNT) bit is copied to the SDA output. When I2CxCNT is not '0', the value of the ACKDT bit is copied to SDA. When I2CxCNT is '0', the value of the ACKCNT bit is copied to SDA. In most applications, the value of ACKDT may be '0', which represents an $\overline{ACK}$ (see figure below).

Figure 2-5. Master $\overline{\text{ACK}}$ (I2CxCNT = 1)



If the SDA line remains at a high logic level during the 9th SCL pulse, this is defined as a Not Acknowledge (NACK) sequence (see figure below).

Figure 2-6. Master NACK (I2CxCNT = 0)



A NACK is generated when any of the following conditions occurs:

- No slave device is present on the bus that owns the transmitted address
- The receiver is busy and is not ready for communication
- The receiver gets data or commands that it cannot understand
- The receiver cannot receive any more data

- A master-receiver has received the requested data and is ready to terminate transmission
- An I²C Error condition has occurred
- The I2CxCNT register has reached a '0' value and ACKCNT is set (ACKCNT = 1).

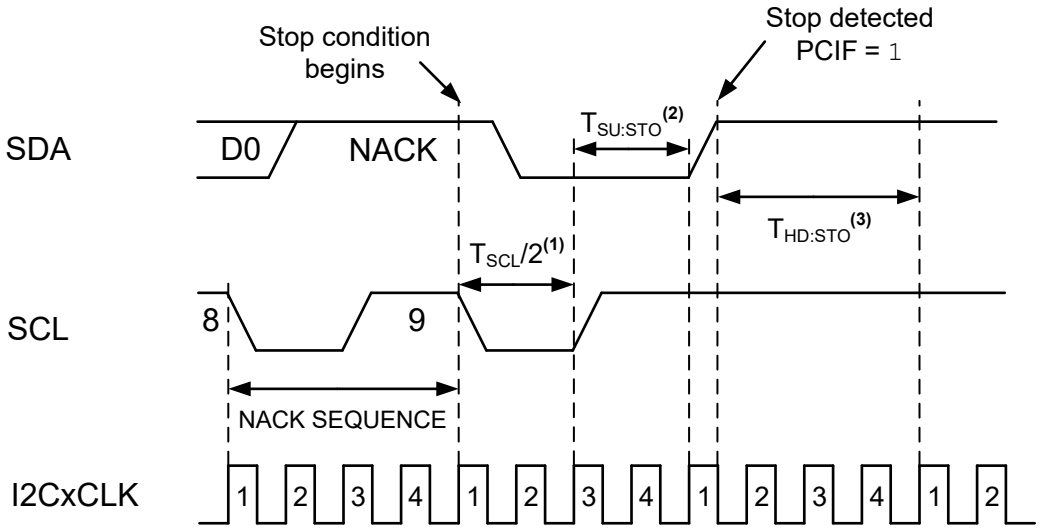
The master device can then decide to either generate a Stop condition to terminate the transfer, or issue a Restart condition to hold the bus and begin a new transfer.

2.8 Stop Condition

The I²C specification defines a Stop condition as the transition of the SDA line from an active state to an idle state while the SCL line is idle. The master will issue a Stop condition when it has completed its transactions and is ready to release control of the bus, or if a bus time-out occurs.

Important: Note that at least one SCL low period must appear before a Stop condition is valid. If the SDA line transitions low and then high again while the SCL line is high, the Stop condition is ignored and a Start/Restart condition will be detected by the receiver (see figure below).

Figure 2-7. Stop Condition



Note:

1. At least one SCL low time must appear before a stop is valid.
2. See device data sheet for Stop condition setup times.
3. See device data sheet for Stop condition hold times.

After the $\overline{ACK}$/NACK sequence of the final byte of the transmitted/received I²C packet, hardware pulls the SCL line low for $T_{SCL}/2$, and then releases SCL. Hardware samples SCL to ensure a logic high level. SDA is then released, and the transition of SDA from low to high while SCL is high causes the Stop Condition Interrupt Flag (PCIF) bit to be set.

2.9 SDA and SCL Pins

The Serial Data (SDA) and Serial Clock (SCL) pins are used by the I²C module to control the I²C bus lines. Unlike previous versions of the MSSP, the SCL and SDA pins must be configured manually in open-drain operation by setting the appropriate bits in the associated port's Open-Drain Control register (ODCON). Also unlike previous versions of the MSSP, the port's Direction Control register (TRIS) must have the SDA and SCL pins configured as outputs by clearing the appropriate TRIS bits. Finally, slew rate control, internal pull-up resistor selection, and input threshold levels for each pin can be configured using the RxyI2C I²C Pad Control register.

Important: Note that previous MSSP modules have recommended using external pull-up resistors rather than the internal weak pull-ups. However, the internal pull-ups located on the dedicated I²C pins may now be used, depending on the bus transmission frequency and capacitance. The internal pull-ups can be configured in the RxyI2C register.

The SDA and SCL pins are typically assigned to two I/O port pins, and must be enabled using the Peripheral Pin Select (PPS) module. The PPS module has two dedicated I²C input registers: I2CxSCLPPS, which defines the SCL input pin, and I2CxDATPPS, which defines the SDA input pin. SDA and SCL outputs are also defined via the PPS module. The outputs use the RxyPPS registers to define the signal the pin will output.

Important: Note that both the SDA and SCL inputs and outputs must be defined, and must be assigned to the same pins. For example, if the SDA pin is assigned to pin RC4, both the I2CxDATPPS and the RC4PPS registers must be mapped to pin RC4. If both input and output signals are not mapped to the same pin, or if one of the signals are not mapped at all, no communication will take place.

The PPS module also allows for alternate pins to be used instead of the default pin locations. If an alternate pin location is desired, simply load the appropriate PPS registers with the new location.

Important: Note that some devices allow digital peripherals to be relocated to any pin, while other devices only allow the digital peripherals to be moved to pins within two I/O ports. Please refer to the device data sheet's PPS chapter for more details. Also, if the I²C pin locations are moved from the default pins, the new locations may not be configured for I²C levels, and would require the open-drain, slew rate, and input threshold control registers to be configured for I²C levels, and the bus may require external pull-up resistors since the weak pull-ups of non-I²C pins are not suitable for communication.

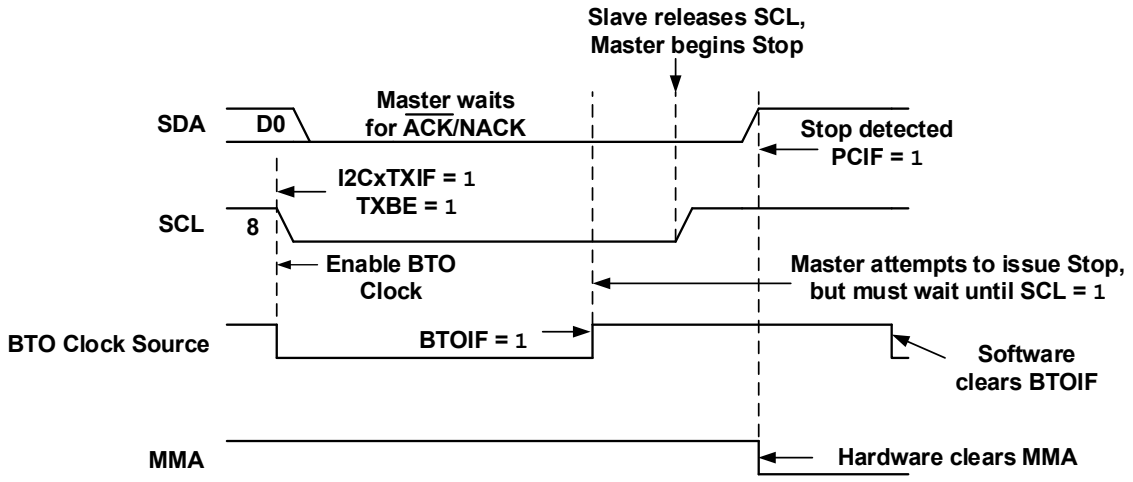
2.10 Bus Time-Out

SMBus and PMBus protocols require a bus watchdog to prevent a stalled device from hanging the bus indefinitely. The I²C module provides a bus time-out feature that can be used to reset the module if one of the bus devices is taking too long to respond. The I2C Bus Time-Out (I2CxBTO) register is used to select the time-out source for the module. When the time-out source expires, the I2CxBTO register notifies the module hardware and resets the module.

If the module is configured as a slave and a bus time-out event occurs while the slave is active (Slave Mode Active bit (SMA) = 1), the SMA and Slave Clock Stretching (CSTR) bits are cleared, the module is reset, and the Bus Time-Out Interrupt Flag (BTOIF) bit is set.

If the module is configured as a master and a bus time-out event occurs while the master is active (Master Mode Active bit (MMA) = 1), the module immediately attempts to transmit a Stop condition and sets BTOIF. Generation of the Stop condition may be delayed if a slave is stretching the clock. The MMA bit is only cleared after a Stop condition has been generated (see figure below).

Figure 2-8. Master Transmit Bus Time-Out Event Example



2.11 Data Byte Count

The data byte count is the number of bytes in a complete I^2C packet. The I2C Byte Count (I2CxCNT) register is used to specify the length, in bytes, of the complete transaction. The value loaded into I2CxCNT will decrement each time a data byte is transmitted or received by the module.

When a byte transfer causes the I2CxCNT register to decrement to '0', the Byte Count Interrupt Flag (CNTIF) bit of the I2CxPIR register is set, and if the Byte Count Interrupt Enable (CNTIE) bit of the I2CxPIE register is set,the general purpose I2C Interrupt Flag (I2CxIF) bit is also set. The I2CxIF is a read-only bit and can only be cleared by clearing all enabled Interrupt Flag bits in the I2CxPIR register.

The I2CxCNT register can be read at any time, but it is recommended that a double read is performed to ensure a valid read.

The I2CxCNT register can be written to, but care is required to prevent register corruption. If the I2CxCNT register is written to during the 8th falling SCL edge during reception, or during the 9th falling SCL edge during transmission, the register value may be corrupted. In Slave mode, I2CxCNT can be safely written to any time the slave is stretching the clock (CSTR = 1), or after a Stop condition has been received. In Master mode, I2CxCNT can be safely written to any time the master state machine is paused (MDR = 1), or when the bus is idle (BFRE = 1). If the I^2C packet is longer than 255 bytes, the I2CxCNT value can be updated mid-message to prevent the count from reaching '0'; however, the preventative measures listed above must be followed.

The I2CxCNT value can be automatically loaded when the Auto-Load I2C Count Register Enable (ACNT) bit of the I2CxCON2 register is set. When ACNT is set, the data byte following the address byte is loaded into I2CxCNT, and the value of the Acknowledge Data (ACKDT) bit is used for the $\overline{ACK}$ response.

When in either Slave-Read or Master-Write mode and the I2CxCNT value is not '0', the value of the ACKDT bit is used for the $\overline{ACK}$ response. When I2CxCNT = 0, the value of the Acknowledge End of Count (ACKCNT) bit is used for the $\overline{ACK}$ response.

When the module is in Master mode and I2CxCNT = 0 and the Restart Enable (RSEN) bit is clear, the master state machine will automatically generate a Stop condition instead of reading/writing another byte. When I2CxCNT = 0 and RSEN = 1, the master will set the Master Data Ready (MDR) bit, stretch the clock, and wait for the Start bit to be set before sending a Restart condition and the address of the slave it wishes to communicate with.

3. Interrupts for Address Match, Transmit Buffer Empty, Receive Buffer Full, Bus Time-Out, Data Byte Count, Acknowledge, and Not Acknowledge

The stand-alone I²C module contains additional interrupt features designed to assist with communication functions. In addition to the MSSP module's Start/Restart Condition (SCIF), Stop Condition (PCIF), Bus Collision (BCLIF), and transmit, receive, and acknowledge (SSPIF) interrupts, the stand-alone I²C module adds an Address Match (ADRIF), Transmit Buffer Empty (TXBE), Receive Buffer Full (RXBF), Bus Time-Out (BTOIF), Data Byte Count (CNTIF), Acknowledge Status Time (ACKTIF), and Not Acknowledge Detect (NACKIF).

The stand-alone I²C module incorporates a new register, the I2C Interrupt Flag register (I2CxPIR), which handles several I²C related interrupts. Additionally, when any of the flag bits in I2CxPIR become set and the associated I2CxPIE Interrupt Enable bit is set, the generic I2C Interrupt Flag (I2CxIF) is also set. If the matching Interrupt Enable bit is set, an interrupt is generated whenever the Interrupt Flag bit is set. If the associated Interrupt Enable bit is clear, the Interrupt flag will still be set when the Interrupt condition occurs, however, no interrupt will be triggered.

Important: Note that the generic I2CxIF bit is read-only and is only cleared by hardware when all bits in the I2CxPIR register are clear.

The I2CxPIR contains the following Interrupt Flag bits:

- CNTIF: Byte Count Interrupt Flag
- ACKTIF: Acknowledge Status Time Interrupt Flag
- WRIF: Data Write Interrupt Flag
- ADRIF: Address Interrupt Flag
- PCIF: Stop Condition Interrupt Flag
- RSCIF: Restart Condition Interrupt Flag
- SCIF: Start Condition Interrupt Flag

The CNTIF becomes set (CNTIF = 1) when the I2CxCNT register value reaches '0', indicating that all bytes in the data frame have been transmitted or received. CNTIF is set after the 9th falling edge of SCL when the I2CxCNT = 0.

The ACKTIF becomes set (ACKTIF = 1) after the 9th falling edge of SCL for any byte when the device is addressed as a slave in any I²C Slave mode or I²C Multi-Master mode whenever an $\overline{\text{ACK}}$ is detected.

The WRIF becomes set (WRIF = 1) after the 8th falling edge of SCL when the module receives a data byte. This bit is only active in any I²C Slave mode or I²C Multi-Master mode. Once the data byte is received, WRIF is set, as is the Receive Buffer Full Status (RXBF) bit, the I2C Receive Interrupt Flag (I2CxRXIF) bit, and if the Data Write Interrupt and Hold Enable (WRIE) bit is set, the generic I2CxIF bit is also set. The WRIF bit is read/write and must be cleared by user software, while the RXBF, I2CxRXIF, and I2CxIF are read-only, and are cleared by reading the I2CxRXB.

The ADRIF becomes set on the 8th falling edge of SCL after the module has received either a matching 7-bit address byte or the matching upper or lower bytes of a 10-bit address. This bit is only active in Slave mode or Multi-Master mode. Upon receiving a matching address byte, the ADRIF bit is set.

The PCIF is set whenever a Stop condition is detected on the bus.

The RSCIF is set upon the detection of a Restart condition.

The SCIF is set upon the detection of a Start condition.

In addition to the I2CxPIR register, the stand-alone module incorporates the I2C Error (I2CxERR) register. The I2CxERR register contains three Interrupt Flag bits that are used to detect bus errors. These bits are read/write and must be cleared by user software. The I2CxERR register also includes the Enable bits for these three functions.

The I2CxERR register contains the following Interrupt Flag bits:

- BTOIF: Bus Time-Out Interrupt Flag
- BCLIF: Bus Collision Detect Interrupt Flag
- NACKIF: NACK Detect Interrupt Flag

BTOIF is set when a bus time-out occurs. The bus time-out time frame is controlled by the I2C Bus Time-Out (I2CxBTO) register. If a bus time-out event occurs and the module is configured as a master and is active (MMA = `1`), the BTOIF is set and the module immediately tries to issue a Stop condition. When BTOIF becomes set, and the associated Bus Time-Out Interrupt Enable (BTOIE) bit is set, the generic I2C Error Interrupt Flag (I2CxEIF) bit is also set. The I2CxEIF bit is read-only, and is cleared by hardware when all error Interrupt Flag bits in the I2CxERR register are clear.

BCLIF is set whenever a bus collision is detected. A bus collision occurs any time the SDA input is sampled low while the both the SDA and SCL outputs are high. When BCLIF is set, and the associated Bus Collision Detect Interrupt Enable (BCLIE) bit is set, the generic I2CxEIF bit is also set.

NACKIF is set when either the master or slave is active (SMA = `1` || MMA = `1`) and a NACK is detected on the bus. A NACK response occurs during the 9th SCL pulse when the SDA line is released high. When the module is in Master mode, a NACK can be issued when the master has finished receiving data from the slave, or in the event it did not receive a byte. In Slave mode, the slave issues a NACK when it does not receive a matching address, or did not receive the last data byte. A NACK can also be automatically sent if any of the following bits are set, which will set NACKIF, and if the NACK Detect Interrupt Enable (NACKIE) bit is set, hardware also sets I2CxEIF:

- TXWE: Transmit Write Error Status bit
- RXRE: Receive Read Error Status bit
- TXU: Transmit Underflow Status bit
- RXO: Receive Overflow Status bit

4. I²C Master Mode Operation

To begin any I²C communication, the master hardware checks to ensure that the bus is in an idle state, which means both the SCL and SDA lines are floating high. Master hardware monitors the Bus Free (BFRE) bit to be set, indicating the bus is idle. The master then transmits a Start condition, followed by the address of the slave it intends to communicate with. The slave address can be either 7-bit or 10-bit, depending on the application design.

In 7-bit Addressing mode, the Least Significant bit (LSb) acts as the Read/not Write (R/$\overline{W}$) bit, while in 10-bit Addressing mode, the LSb of the address high byte is considered the R/$\overline{W}$ bit. When the R/$\overline{W}$ bit is set, the master intends to read from the slave. If the R/$\overline{W}$ bit is cleared, the master intends to write to the slave. If the addressed slave device exists on the bus, it must respond with an Acknowledge ($\overline{ACK}$) sequence.

The master then continues to either receive data from the slave, write data to the slave, or a combination of both. Data is always transmitted Most Significant bit (MSb) first. When the master intends to halt further transmission, it transmits a Stop condition, signaling to the slave that communication is to be terminated, or a Restart condition, signaling the bus that the current master wishes to hold the bus to communicate with the same or other slaves.

Master mode is selected by configuring the MODE<2:0> bits of the I2CxCON0 register. There are four Master mode configurations:

- I²C Master mode with 7-bit address
- I²C Master mode with 10-bit address
- I²C Multi-Master – Master mode with 7-bit address and Slave mode with two 7-bit addresses with masking
- I²C Multi-Master – Master mode with 7-bit address and Slave mode with four 7-bit addresses.

The master device generates the SCL pulses, as well as the Start, Restart, and Stop conditions. Transmission always begins with a Start condition, and can end with either a Stop condition or Restart condition. When the master has completed all transactions, and is ready to release the bus, it will generate a Stop condition. If the master wishes to stop communicating with one slave, but wants to hold the bus to address another slave, it issues a Restart condition. Control of the bus can only be asserted when the Bus Free (BFRE) bit of the I2CxSTAT0 register is set.

The steps to initiate a transaction depend on the settings of the Address Buffer Disable (ABD) bit of the I2CxCON2 register.

When the ABD bit is clear (ABD = 0), the address buffer registers, I2CxADB0 and I2CxADB1, are active and used for slave address transmission. The module will automatically load the transmit shift register with an address stored in one of the address buffers. Software must set the Start (S) bit to initiate communication with the slave.

When the ABD bit is set (ABD = 1), the address buffers are inactive and ignored for transmission. In this case, user software must load the I2CxTXB with the slave address to begin communication, and any writing to the Start bit will be ignored.

4.1 Master Clock Timing

The I²C module clock is generated by module hardware in Master mode. The I2CxCLK register provides the clock source for the module, which can be selected from several peripherals. Master clock timing is

controlled by the Fast Mode Enable (FME) bit of the I2CxCON2 register. The FME bit controls the number of times the SCL pin is sampled before the master hardware drives it.

Important: Note that the I²C clock is not the same as the SCL, rather it is used to time the SCL output.

The clock source selected by I2CxCLK, in combination with the FME bit, is used by master hardware to time the SCL signal. For example, if the Medium Frequency Internal Oscillator (MFINTOSC), which generates a 500 kHz output, is selected as the I²C clock source, the SCL frequency would not be 500 kHz. The MFINTOSC signal would be divided by either 4 or 5, depending on the value of the FME bit (see equations below).

When the FME bit is cleared, one SCL period (T_{SCL}) consists of five clock periods of the I²C clock input source selected by the I2CxCLK register (see figure below). The first clock period is used to drive SCL low, and the second clock period samples SCL to ensure it is in fact low. The third clock period releases SCL high, and the fourth and fifth clock periods sample the SCL to detect if the SCL pin is indeed high or if the slave is stretching the clock.

If the slave is stretching the clock, module hardware waits, checking each successive I²C clock period until the hardware detects a high level on SCL. Once the high level is detected, hardware uses the next two successive I²C clock periods to verify the SCL is high.

Equation 4-1. SCL Frequency Example (FME = 0)

When FME = 0

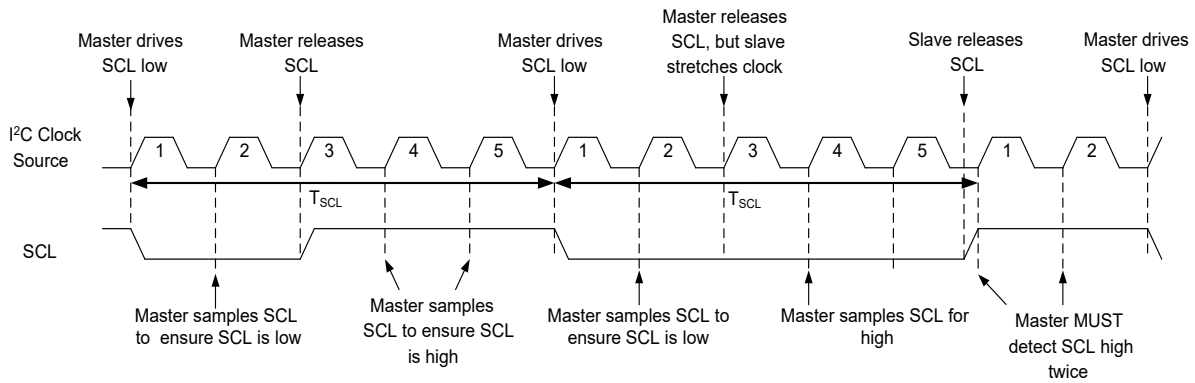
$$f_{SCL} = \frac{f_{I2CCLK}}{5}$$

Example:

- I2CxCLK = MFINTOSC (500 kHz)
- FME = 0

$$f_{SCL} = \frac{500kHz}{5} = 100\,kHz$$

Figure 4-1. I²C SCL Timing (FME = 0)



When the FME bit is set, one SCL period (T_{SCL}) consists of four clock periods of the I²C clock input source selected by the I2CxCLK register (see figure below). The first clock period drives SCL low, and the second clock period samples SCL to ensure it is low. The third clock period causes the master to release the SCL, driving SCL high. The fourth clock period samples SCL to determine whether it is high or being stretched by a slave. If the slave is stretching the clock, module hardware waits, checking each successive I²C clock period until the hardware detects a high level on SCL. Once the high level is detected, hardware uses the next successive I²C clock period to verify if the SCL is high.

Equation 4-2. SCL Frequency Example (FME = 1)

When FME = 1

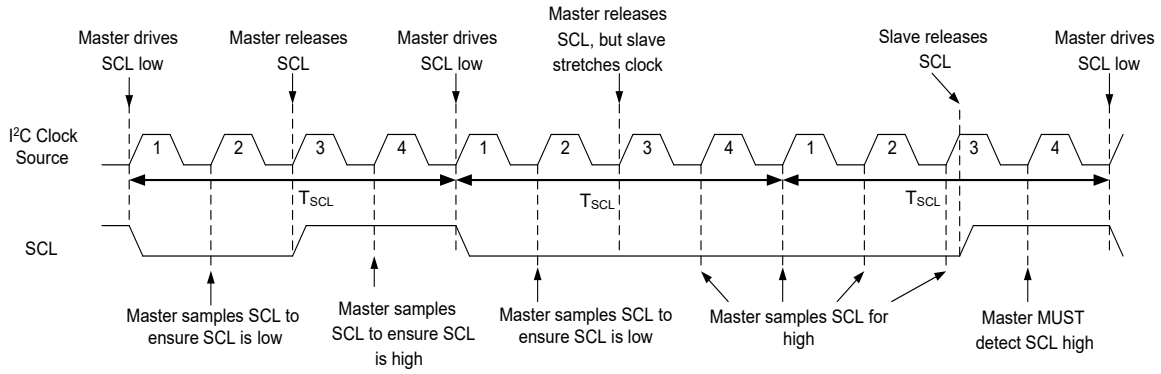
$$f_{SCL} = \frac{f_{I2CCLK}}{4}$$

Example:

- I2CxCLK = MFINTOSC (500 kHz)
- FME = 1

$$f_{SCL} = \frac{500kHz}{4} = 125\,kHz$$

Figure 4-2. I²C SCL Timing (FME = 1)



5. Bus Free Time

The Bus Free (BFRE) bit of the I2CxSTAT0 register is used to indicate the status of the bus. Master hardware sets this bit when it detects an idle bus. When BFRE = 1, any master device on the bus can compete for control of the bus. When BFRE = 0, the bus is considered busy, and any attempts by a master to control the bus will cause a collision. The Bus Free Time (BFRET) bits of the I2CxCON1 register are used to select the number of I^2C clock pulses that delay the master hardware from setting the BFRE bit after the bus is detected in the idle state. The BFRET bits are used to ensure that module meets the minimum stop hold time defined by the I^2C specification.

6. Master Mode Configuration and Operation

The steps listed below can be used to configure the I²C module for Master mode operation.

6.1 Initialization

To begin I²C Master mode communication, the following register bits must be properly configured during initialization (see code example below):

I²C Initialization Example

```
static i2c_error lastError = I2C1_GOOD;

void I2C1_Initialize(void)               // Initialize I2C Module
{
    if(!I2C1CON0bits.EN || lastError != I2C1_GOOD)
    {
        lastError = I2C1_GOOD;
        I2C1CON0 = 0x04;                 // Master 7-bit address mode
        I2C1CON1 = 0x80;                 // ACKDT = ACK, ACKCNT = NACK
        I2C1CON2 = 0x24;                 // Enable Address Buffers
                                         // BFRET = 8 I2C pulses
                                         // FME = 1
        I2C1CLK = 0x03;                  // MFINTOSC (500 kHz)
        I2C1PIR = 0;                     // Clear all interrupt flags
        I2C1ERR = 0;                     // Clear all error flags
        I2C1CON0bits.EN = 1;             // Enable I2C module
    }
}

void PIN_MANAGER_Initialize(void)        // Initialize SCL and SDA pins
{
    LATC = 0x00;                         // Clear PORTC write latches
    TRISC = 0xE7;                        // RC3, RC4 initialized as outputs
    ANSELC = 0xE7;                       // Clear RC3, RC4 analog input
    ODCONC = 0x18;                       // Must configure RC3, RC4 as OD
    RC3I2C = 0x01;                       // Standard GPIO slew rate
                                         // Internal pull-ups not used
                                         // I2C specific thresholds
    SLRCONCbits.SLRC3 = 0;               // No slew rate limiting
    RC4I2C = 0x01;
    SLRCONCbits.SLRC4 = 0;

    // PPS configuration
    bool state = (unsigned char)GIE;
    GIE = 0;
    PPSLOCK = 0x55;                      // Unlock sequence
    PPSLOCK = 0xAA;
    PPSLOCKbits.PPSLOCKED = 0x00;        // unlock PPS
    RC3PPS = 0x21;                       // RC3->I2C1:SCL1;
    RC4PPS = 0x22;                       // RC4->I2C1:SDA1;
    I2C1SDAPPSbits.I2C1SDAPPS = 0x14;    // RC4->I2C1:SDA1;
    I2C1SCLPPSbits.I2C1SCLPPS = 0x13;    // RC3->I2C1:SCL1;
    PPSLOCK = 0x55;                      // Lock sequence
    PPSLOCK = 0xAA;
    PPSLOCKbits.PPSLOCKED = 0x01;        // lock PPS
    GIE = state;
}
```

I2CxCON0: The I2CxCON0 contains the module Enable (EN) bit and the Mode Select (MODE<2:0>) bits. The MODE<2:0> bits are used to select the Communications mode, and the EN bit enables the Master state machine hardware. MODE<2:0> bit settings must not be changed while the EN bit is set (module is enabled).

I2CxCON1: The I2CxCON1 register contains the Acknowledge End of Count (ACKCNT) and Acknowledge Data (ACKDT) bits.

The ACKCNT bit reflects the value transmitted after the I2CxCNT register has reached '0', signaling the end of the packet. When ACKCNT is clear, the module will issue an $\overline{ACK}$; when set, the module issues a NACK. This bit can be modified during run time, but must only be changed before the I2CxCNT reaches '0' and before an Acknowledge sequence is issued. If there are errors in either the I2CxERR or I2CxSTAT registers, master hardware automatically overrides this bit setting and generates a NACK.

The ACKDT bit reflects the value transmitted after a matching address is received, or after a byte is received while I2CxCNT is not '0'. When ACKDT is clear, an $\overline{ACK}$ is issued; when ACKDT is set, a NACK is issued. The ACKDT bit value can be modified during run time, but must only be changed before an Acknowledge sequence is issued. If there are errors in either the I2CxERR or I2CxSTAT registers, master hardware automatically overrides this bit setting and generates a NACK.

I2CxCON2: The I2CxCON2 register holds the Auto-Load I2C Count Register Enable (ACNT), Fast Mode Enable (FME), Address Buffer Disable (ABD), SDA Hold Time Selection (SDAHT<1:0>), and Bus Free Time Selection (BFRET<1:0>) bits.

The ACNT bit enables/disables the auto-loading of the I2CxCNT register. Auto-loading of I2CxCNT can be useful when a slave device does not know the size of the data packet, or when the master needs to change the size of the packet to transmit. When ACNT is set, the first byte following the matching address is used as the value that is loaded into the I2CxCNT register. For example, if the master device intends to transmit three data bytes to a slave, the byte following the address would have a value of '3', and would be loaded into the master device's I2CxCNT register during transmission. When the byte is received by the slave device, it is loaded into the slave's I2CxCNT register. Of course, this assumes that both the master and the slave have the I2CxCNT register feature available, and both devices have ACNT set.

The FME bit is used in combination with the I2CxCLK register to determine the SCL frequency. When FME is set, one SCL period consists of four clock periods of the I2CxCLK clock source. When FME is clear, one SCL period consists of five clock periods of the I2CxCLK source.

The ABD bit enables/disables the use of the dedicated Address Buffer registers. In Master mode, the address intended to be transmitted to the slave can be loaded into the I2CxADB0/1 registers.

When ABD = 1, the I2CxADB0/1 registers are ignored, and the slave address must be loaded into the I2CxTXB transmit buffer by user software to initiate communication. Writing to the Start bit is ignored.

When ABD = 0, the address data stored in I2CxADB0/1 is loaded into the transmit shift register automatically after a Start condition is issued by user software.

The SDAHT<1:0> bits are used to configure the amount of time the SDA line is held valid after the falling edge of SCL. The SDAHT<1:0> bits may be configured based on the bus capacitance; buses with larger capacitance may need longer hold times to ensure valid data.

The BFRET<1:0> bits are used to select the amount of I^2C clock cycles used to delay hardware from setting the BFRE bit. The BFRET<1:0> bits can be used to meet the minimum stop hold time as defined by the I^2C specification. Note that in systems with more than one master, it is possible that the BFRE bit may never become set if another master device takes control of the bus before the BFRE bit becomes set. In this case, care must be used when selecting the BFRET<1:0> timing.

I2CxCLK: The I2CxCLK register selects the I^2C clocking source, and is used in combination with the FME bit to determine the SCL frequency. Some source selections, such as a timer, must also be configured and enabled during initialization.

Important: Note that not all I2CxCLK selections can be used to achieve valid SCL frequencies. For example, if a 400 kHz SCL frequency is desired, the HFINTOSC source may not be a feasible selection. The HFINTOSC may be configured to operate at 16 MHz. If the FME bit is set, the SCL frequency would be the HFINTOSC frequency divided by 4, or 4 MHz. If the FME bit is clear, the SCL frequency would be the HFINTOSC frequency divided by 5, or 3.2 MHz.

I2CxBTO: The I2CxBTO register selects the timing source used for the Bus Time-Out feature. The current time-out sources are either a CLC or a Timer, and those modules must also be configured during initialization. The time-out source must be configured such that a device does not stall the bus for too long, but doesn't interfere with timely data processing or clock stretching.

I2CxERR: The I2CxERR register contains the Bus Time-Out Interrupt Enable (BTOIE), the Bus Collision Detect Interrupt Enable (BCLIE), and NACK Detect Interrupt Enable (NACKIE) bits. If these interrupts are not needed by the application, this register does not need to be explicitly initialized.

I2CxCNT: The I2CxCNT register is loaded with the number of data bytes present in a I^2C packet. The I2CxCNT can be loaded during initialization or run time directly, but it is recommended to write to this register only if the module is idle or during clock stretching. Writing at any other time may corrupt the register. I2CxCNT can also be automatically loaded during run time when the ACNT bit of I2CxCON2 is set. In this case, the first byte following the address byte(s) is loaded into I2CxCNT by module hardware. The I2CxCNT value must only include the number of data bytes in the packet, and not any address bytes.

I2CxPIE: The I2CxPIE register contains several I^2C specific Interrupt Enable bits. Initialization is only required if one or more of the following interrupts are necessary:

- Byte Count Interrupts
- Acknowledge Interrupt and Hold
- Data Write Interrupt and Hold
- Address Interrupt and Hold
- Stop Condition Interrupts
- Restart Condition Interrupts
- Start Condition Interrupts

I2CxADB0: The I2CxADB0 register initialization is only required when using 10-bit address Master mode and the ABD bit is clear. In this case, the lower byte of the 10-bit address is loaded into I2CxADB0 and copied into the transmit shift register upon the issue of a Start condition.

I2CxADB1: The I2CxADB1 initialization is required when using 7-bit or 10-bit address Master modes and the ABD bit is clear. In 7-bit address Master mode, the I2CxADB1 register holds the 7-bit slave address and R/$\overline{W}$ bit, and I2CxADB0 is ignored. In 10-bit address Master mode, I2CxADB1 holds the higher byte of the 10-bit address. The five most significant bits of I2CxADB1 are defined as a constant '11110' value by the I^2C specification, and have to be included in the upper address byte. This constant value is followed by bits '10' and '9' of the 10-bit address, and finally the R/$\overline{W}$ bit.

RxyI2C: The RxyI2C register controls the I^2C specific I/O pads. Most PIC devices dedicate one or two pairs of I/O pins to the I^2C module. The RxyI2C register is used to configure the pin slew rate, input threshold level, and internal pull-up configurations.

The SLEW bit controls the slew rate. When SLEW is set, I^2C specific slew rate is enabled, which overrides the standard pin slew rate limiting, and the SLRxCON bit associated with the pin is ignored. When SLEW is clear, the module uses the standard pad slew rate, which is enabled/disabled via the

SLRxCON bit associated with the pin. Lower bus speeds may not need any slew rate limiting, while buses with higher speeds may need slew rate limiting.

The TH<1:0> bits control the I2C input threshold level. These bits can be configured to SMBus 3.0, SMBus 2.0, I2C specific, or standard I/O input threshold levels to meet the specific protocol requirements. When either the SMBus 3.0, SMBus 2.0, or I2C specific levels are selected, the INLVL bit associated with the pin is ignored. If standard I/O threshold levels are selected, the INLVL bit associated with the pin can be configured for either ST or TTL logic levels.

The PU<1:0> bits are used to select the internal pull-up drive strength. The PU<1:0> bits can be configured to increase the current drive of the pull-up, making the internal pull-ups strong enough to be used instead of external pull-up resistors. If external pull-ups are to be used in the application, the PU<1:0> bits can be configured for standard weak pull-ups, which can be enabled/disabled via the WPU bit associated with the pin.

TRISx registers: The TRIS registers provide I/O direction support to PORT pins. When using the I2C module, the TRIS bits associated with the SDA and SCL pins must be initialized clear (TRISxy = 0). All previous I2C module designs required the TRIS bits to be set. During run time, direction control is handled by module hardware.

ODCONx: The I2C module uses an open-drain circuit configuration. The ODCONx bits associated with the SCL and SDA pins must be configured for open-drain (ODCONxy = 1).

I2C PPS registers: The Peripheral Pin Select (PPS) feature allows digital signals to be moved from their default pin location to another location. To enable a digital peripheral's input and/or output signals, the appropriate PPS registers must be configured. When using the I2C module, both the input PPS and output PPS registers must be configured due to the bidirectional nature of the I2C bus. Both the input and output PPS registers for each I2C signal must be routed to the same pin. In other words, if the I2CxSCLPPS input register is mapped to pin RC3, the RC3PPS register must also be mapped to pin RC3.

Input configuration is handled by the I2CxSCLPPS and I2CxSDAPPS registers. These registers must be mapped to the desired pins to enable the pin input drivers. Output configuration is handled by the RxyPPS registers. The 'xy' in the register name is a placeholder for the actual port and pin number. For example, If the SDA line is mapped to port pin RC4, the correct register name is RC4PPS. The PPS output registers must also be mapped to the desired pins to enable the pin output driver.

The PPS feature allows the I2C pins to be moved from their default locations, but additional steps must be considered. The default I2C pins use the RxyI2C register to define the slew rate, pull-up configuration, and input threshold levels. If the default pin locations are not used, additional registers, such as INLVLx, WPUx, and SLRCONx must also be configured.

7. Master Mode Transmission

The following section describes the sequence of events when using the I^2C in Master mode transmission.

1. Depending on the configuration of the Address Buffer Disable (ABD) bit, one of two methods is used to begin communication.
   When ABD is clear, the address buffers are enabled. In 7-bit Addressing mode, the 7-bit slave address is loaded into the I2CxADB1 register with the R/$\overline{W}$ bit clear. In 10-bit Addressing mode, the high address byte is loaded into the I2CxADB1 register with the R/$\overline{W}$ bit clear, and the low address byte is loaded into the I2CxADB0 register. The number of data bytes to be transmitted in one packet is loaded into the I2CxCNT register, and the first byte of data is loaded into the I2CxTXB transmit register. After these registers are loaded, master software must set the Start bit to begin communication. Master hardware must wait for BFRE to be set before transmitting the Start condition to avoid bus collisions.

   When ABD is set, the address buffers are disabled. In this case, the number of data bytes to be transmitted in one packet must be loaded into the I2CxCNT register before loading the transmit buffer. In 7-bit Addressing mode, the slave address is loaded into I2CxTXB with the R/$\overline{W}$ bit clear. Writing to the I2CxTXB register will automatically issue a Start condition via module hardware once the BFRE is set. In 10-bit Addressing mode, the slave's high address byte with the R/$\overline{W}$ bit clear is loaded into the I2CxTXB register. Once the BFRE bit is set, module hardware shifts out the high address byte. In both 7-bit and 10-bit Addressing modes, when ABD is set, writes to the Start bit are ignored.
2. Master hardware waits for the BFRE bit to be set, then shifts out the Start condition. Module hardware sets the Master Mode Active (MMA) and Start Condition Interrupt Flag (SCIF) bits.
3. Master transmits either the 7-bit slave address with R/$\overline{W}$ clear or the 10-bit high address byte with R/$\overline{W}$ clear.
   In 7-bit mode, if the transmit buffer is empty (TXBE = 1), the I2CxCNT register is not '0', and the CSD bit is clear, the I2CxTXIF and MDR bits are set, and the clock will be stretched by master hardware, allowing master software to write new data into I2CxTXB. Once I2CxTXB has been written, master hardware releases SCL and waits for an $\overline{ACK}$/NACK sequence to be shifted in from the slave.

   In 10-bit mode, module hardware waits for the $\overline{ACK}$/NACK from the slave. If a NACK is received, module hardware immediately issues a Stop condition. If an $\overline{ACK}$ is received, module hardware shifts out the 10-bit address low byte. If TXBE is set, and I2CxCNT is not '0', the I2CxTXIF and MDR bits are set, and SCL is stretched on the 8th falling SCL edge to allow the master to load new data into I2CxTXB. Once I2CxTXB has been written, master hardware releases SCL and waits for an $\overline{ACK}$/NACK sequence to be shifted in from the slave.
4. Master hardware clock out the 9th SCL pulse and waits for the $\overline{ACK}$ response from the slave. If the master receives a NACK, master hardware will issue a Stop condition.
5. If the master receives an $\overline{ACK}$, module hardware transfers the data byte currently in the transmit buffer into the transmit shift register, and the value of I2CxCNT is decremented by one.
6. Master hardware checks to see if I2CxCNT is '0'.
   If I2CxCNT is not '0', go back to step 5. If I2CxCNT is '0', and ABD is clear,master hardware issues a Stop condition, or sets the MDR bit if the RSEN bit is set and waits for master software to set the Start bit again to issue a Restart condition.

   If I2CxCNT is '0' and the ABD bit is set, hardware issues a Stop condition, or sets the MDR bit if the RSEN bit is also set and waits for software to load the I2CxTXB register with new address data.

8. Master Mode Reception

The following section describes the sequence of events when using the I^2C in Master mode reception.

1. Depending on the configuration of the Address Buffer Disable (ABD) bit, one of two methods is used to begin communication.
   When ABD is clear, the address buffers are enabled. In 7-bit Addressing mode, the 7-bit slave address is loaded into the I2CxADB1 register with the R/$\overline{W}$ bit clear. In 10-bit Addressing mode, the high address byte is loaded into the I2CxADB1 register with the R/$\overline{W}$ bit set, and the low address byte is loaded into the I2CxADB0 register. The number of data bytes to be transmitted in one packet is loaded into the I2CxCNT register, and the first byte of data is loaded into the I2CxTXB transmit register. After these registers are loaded, master software must set the Start bit to begin communication. Master hardware must wait for BFRE to be set before transmitting the Start condition to avoid bus collisions.

   When ABD is set, the address buffers are disabled. In this case, the number of data bytes to be transmitted in one packet must be loaded into the I2CxCNT register before loading the transmit register. In 7-bit Addressing mode, the slave address is loaded into I2CxTXB with the R/$\overline{W}$ bit set. Writing to the I2CxTXB register will automatically issue a Start condition via module hardware once the BFRE is set. In 10-bit Addressing mode, the slave's high address byte with the R/$\overline{W}$ bit clear is loaded into the I2CxTXB register. Once the BFRE bit is set, module hardware shifts out the high address byte. In both 7-bit and 10-bit Addressing modes, when ABD is set, writes to the Start bit are ignored.
2. Master hardware waits for the BFRE bit to be set, then shifts out the Start condition. Module hardware sets the Master Mode Active (MMA) and Start Condition Interrupt Flag (SCIF) bits.
3. Master transmits either the 7-bit slave address with R/$\overline{W}$ set or the 10-bit high address byte with R/$\overline{W}$ set.
   In 10-bit mode, module hardware waits for the $\overline{ACK}$/NACK from the slave. If a NACK is received, module hardware immediately issues a Stop condition. If an $\overline{ACK}$ is received, module hardware shifts out the 10-bit address low byte.
4. Master hardware monitors the SDA line to determine if a slave is stretching the clock, and waits until the SDA line is sampled high.
5. Master hardware transmits the 9th clock pulse, clocking in the slave's $\overline{ACK}$/NACK response.
6. If the master receives an $\overline{ACK}$, hardware clocks the data byte from the slave into the shift register. If the master receives a NACK, and the ABD bit is clear, master hardware generates a Stop condition, or sets the MDR bit if RSEN is also set and waits for software to set the Start bit to generate a Restart condition.

   If the master receives a NACK and the ABD bit is set, master hardware generates a Stop condition, or sets the MDR bit if RSEN is also set and waits for software to load new address data into I2CxTXB. Software writes to the Start bit are ignored.
7. If the previous data is still in the I2CxRXB register (RXBF = `1`) when the first 7 bits of the new byte is received into the shift register, the MDR bit is set, and the clock is stretched after the 7th falling edge of SCL. This allows master software to read I2CxRXB, which clears the RXBF bit, and prevents a receive buffer overflow. Once the RXBF bit is clear, hardware releases SCL.
8. Master hardware clocks in the 8th bit of the new data byte into the shift register, then transfers the complete byte into I2CxRXB, sets the I2CxRXIF and the RXBF bits. I2CxCNT is decremented by one.
9. Master hardware checks I2CxCNT for a '`0`' value.

If I2CxCNT is not ‘0’, hardware transmits the value of the Acknowledge Data (ACKDT) bit as the $\overline{\text{ACK}}$ value to the slave. Master hardware will then continue receive data into the shift register, repeating steps 7-9 until I2CxCNT is ‘0’. It is up to the user to configure the ACKDT bit appropriately. In most cases, the ACKDT bit would have to be clear, so that the slave receives an $\overline{\text{ACK}}$ (logic low level on SDA during the 9th SCL pulse).

If I2CxCNT is ‘0’, hardware transmits the value of the Acknowledge End of Count (ACKCNT) bit as the $\overline{\text{ACK}}$ value to the slave. It is up to the user to properly define the ACKCNT bit. In most cases, this bit is set, indicating a NACK condition. When master hardware detects the NACK on the bus, hardware will also generate a Stop condition. If the ACKCNT bit is clear, an $\overline{\text{ACK}}$ will be issued, and hardware will not automatically generate the Stop condition.

9. External Pull-up Resistor Selection

The I^2C specification proposes two methods to determine the correct pull-up resistor size.

The first method calculates the maximum pull-up resistor size as a function of bus capacitance and rise time (see Equation 9-1). Bus capacitance is the total capacitance of the bus wires/traces, bus connection points, and bus pins, all of which must be considered when calculating the total bus capacitance. Rise time is the period in which the signal transitions from $\mathbf{V_{IL(MAX)}}$ ($0.3*V_{DD}$) to $\mathbf{V_{IH(MIN)}}$($0.7*V_{DD}$). Rise time values are typically located in the device's data sheet.

Bus capacitance would have to be measured to achieve the most accurate pull-up values, but an estimated value, or the maximum allowable capacitance as defined by the I^2C specification, may also be used. The maximum allowable bus capacitance is specified to limit rise time decreases and allow operation at the rated frequency. The bus may operate at higher than allowable bus capacitance levels but at a lower frequency.

Equation 9-1. Maximum Pull-up Resistor Size

$$Rp_{(\max)} = \frac{t_{rise}}{0.8473 * C_{bus}}$$

$\mathbf{R_{p(max)}}$ = Maximum pull-up value

$\mathbf{t_{rise}}$ = Maximum rise time

$\mathbf{C_{bus}}$ = Total bus capacitance

The second method calculates the minimum pull-up resistor size as a function of V_{DD} (see Equation 9-2). The supply voltage limits the minimum resistor value due to the specified minimum sink current of 3 mA for Standard mode (100 kHz) or Fast mode (400 kHz), or 20 mA for Fast mode Plus (1 MHz).

Equation 9-2. Minimum Pull-up Resistor Size

$$Rp_{(\min)} = \frac{V_{DD} - VOL_{(\max)}}{I_{OL}}$$

$\mathbf{R_{p(min)}}$ = Minimum pull-up value

$\mathbf{V_{DD}}$ = Supply voltage

$\mathbf{V_{OL(max)}}$ = Maximum output low voltage

$\mathbf{I_{OL}}$ = Minimum sink current

10. Conclusion

This technical brief has covered the stand-alone I^2C module in Master mode configuration. For more information, please visit www.microchip.com. For code examples, please visit www.microchip.com/mplab/mplab-xpress.

The Microchip Web Site

Microchip provides online support via our web site at http://www.microchip.com/. This web site is used as a means to make files and information easily available to customers. Accessible by using your favorite Internet browser, the web site contains the following information:

- **Product Support** – Data sheets and errata, application notes and sample programs, design resources, user's guides and hardware support documents, latest software releases and archived software
- **General Technical Support** – Frequently Asked Questions (FAQ), technical support requests, online discussion groups, Microchip consultant program member listing
- **Business of Microchip** – Product selector and ordering guides, latest Microchip press releases, listing of seminars and events, listings of Microchip sales offices, distributors and factory representatives

Customer Change Notification Service

Microchip's customer notification service helps keep customers current on Microchip products. Subscribers will receive e-mail notification whenever there are changes, updates, revisions or errata related to a specified product family or development tool of interest.

To register, access the Microchip web site at http://www.microchip.com/. Under "Support", click on "Customer Change Notification" and follow the registration instructions.

Customer Support

Users of Microchip products can receive assistance through several channels:

- Distributor or Representative
- Local Sales Office
- Field Application Engineer (FAE)
- Technical Support

Customers should contact their distributor, representative or Field Application Engineer (FAE) for support. Local sales offices are also available to help customers. A listing of sales offices and locations is included in the back of this document.

Technical support is available through the web site at: http://www.microchip.com/support

Microchip Devices Code Protection Feature

Note the following details of the code protection feature on Microchip devices:

- Microchip products meet the specification contained in their particular Microchip Data Sheet.
- Microchip believes that its family of products is one of the most secure families of its kind on the market today, when used in the intended manner and under normal conditions.
- There are dishonest and possibly illegal methods used to breach the code protection feature. All of these methods, to our knowledge, require using the Microchip products in a manner outside the operating specifications contained in Microchip's Data Sheets. Most likely, the person doing so is engaged in theft of intellectual property.
- Microchip is willing to work with the customer who is concerned about the integrity of their code.

- Neither Microchip nor any other semiconductor manufacturer can guarantee the security of their code. Code protection does not mean that we are guaranteeing the product as “unbreakable.”

Code protection is constantly evolving. We at Microchip are committed to continuously improving the code protection features of our products. Attempts to break Microchip’s code protection feature may be a violation of the Digital Millennium Copyright Act. If such acts allow unauthorized access to your software or other copyrighted work, you may have a right to sue for relief under that Act.

Legal Notice

Trademarks

ISBN: 978-1-5224-4228-8

Quality Management System Certified by DNV

ISO/TS 16949

Microchip received ISO/TS-16949:2009 certification for its worldwide headquarters, design and wafer fabrication facilities in Chandler and Tempe, Arizona; Gresham, Oregon and design centers in California and India. The Company's quality system processes and procedures are for its PIC® MCUs and dsPIC® DSCs, KEELOQ® code hopping devices, Serial EEPROMs, microperipherals, nonvolatile memory and analog products. In addition, Microchip's quality system for the design and manufacture of development systems is ISO 9001:2000 certified.

Worldwide Sales and Service

AMERICAS	ASIA/PACIFIC	ASIA/PACIFIC	EUROPE
Corporate Office 2355 West Chandler Blvd. Chandler, AZ 85224-6199 Tel: 480-792-7200 Fax: 480-792-7277 Technical Support: http://www.microchip.com/support Web Address: www.microchip.com **Atlanta** Duluth, GA Tel: 678-957-9614 Fax: 678-957-1455 **Austin, TX** Tel: 512-257-3370 **Boston** Westborough, MA Tel: 774-760-0087 Fax: 774-760-0088 **Chicago** Itasca, IL Tel: 630-285-0071 Fax: 630-285-0075 **Dallas** Addison, TX Tel: 972-818-7423 Fax: 972-818-2924 **Detroit** Novi, MI Tel: 248-848-4000 **Houston, TX** Tel: 281-894-5983 **Indianapolis** Noblesville, IN Tel: 317-773-8323 Fax: 317-773-5453 Tel: 317-536-2380 **Los Angeles** Mission Viejo, CA Tel: 949-462-9523 Fax: 949-462-9608 Tel: 951-273-7800 **Raleigh, NC** Tel: 919-844-7510 **New York, NY** Tel: 631-435-6000 **San Jose, CA** Tel: 408-735-9110 Tel: 408-436-4270 **Canada - Toronto** Tel: 905-695-1980 Fax: 905-695-2078	**Australia - Sydney** Tel: 61-2-9868-6733 **China - Beijing** Tel: 86-10-8569-7000 **China - Chengdu** Tel: 86-28-8665-5511 **China - Chongqing** Tel: 86-23-8980-9588 **China - Dongguan** Tel: 86-769-8702-9880 **China - Guangzhou** Tel: 86-20-8755-8029 **China - Hangzhou** Tel: 86-571-8792-8115 **China - Hong Kong SAR** Tel: 852-2943-5100 **China - Nanjing** Tel: 86-25-8473-2460 **China - Qingdao** Tel: 86-532-8502-7355 **China - Shanghai** Tel: 86-21-3326-8000 **China - Shenyang** Tel: 86-24-2334-2829 **China - Shenzhen** Tel: 86-755-8864-2200 **China - Suzhou** Tel: 86-186-6233-1526 **China - Wuhan** Tel: 86-27-5980-5300 **China - Xian** Tel: 86-29-8833-7252 **China - Xiamen** Tel: 86-592-2388138 **China - Zhuhai** Tel: 86-756-3210040	**India - Bangalore** Tel: 91-80-3090-4444 **India - New Delhi** Tel: 91-11-4160-8631 **India - Pune** Tel: 91-20-4121-0141 **Japan - Osaka** Tel: 81-6-6152-7160 **Japan - Tokyo** Tel: 81-3-6880- 3770 **Korea - Daegu** Tel: 82-53-744-4301 **Korea - Seoul** Tel: 82-2-554-7200 **Malaysia - Kuala Lumpur** Tel: 60-3-7651-7906 **Malaysia - Penang** Tel: 60-4-227-8870 **Philippines - Manila** Tel: 63-2-634-9065 **Singapore** Tel: 65-6334-8870 **Taiwan - Hsin Chu** Tel: 886-3-577-8366 **Taiwan - Kaohsiung** Tel: 886-7-213-7830 **Taiwan - Taipei** Tel: 886-2-2508-8600 **Thailand - Bangkok** Tel: 66-2-694-1351 **Vietnam - Ho Chi Minh** Tel: 84-28-5448-2100	**Austria - Wels** Tel: 43-7242-2244-39 Fax: 43-7242-2244-393 **Denmark - Copenhagen** Tel: 45-4450-2828 Fax: 45-4485-2829 **Finland - Espoo** Tel: 358-9-4520-820 **France - Paris** Tel: 33-1-69-53-63-20 Fax: 33-1-69-30-90-79 **Germany - Garching** Tel: 49-8931-9700 **Germany - Haan** Tel: 49-2129-3766400 **Germany - Heilbronn** Tel: 49-7131-67-3636 **Germany - Karlsruhe** Tel: 49-721-625370 **Germany - Munich** Tel: 49-89-627-144-0 Fax: 49-89-627-144-44 **Germany - Rosenheim** Tel: 49-8031-354-560 **Israel - Ra'anana** Tel: 972-9-744-7705 **Italy - Milan** Tel: 39-0331-742611 Fax: 39-0331-466781 **Italy - Padova** Tel: 39-049-7625286 **Netherlands - Drunen** Tel: 31-416-690399 Fax: 31-416-690340 **Norway - Trondheim** Tel: 47-72884388 **Poland - Warsaw** Tel: 48-22-3325737 **Romania - Bucharest** Tel: 40-21-407-87-50 **Spain - Madrid** Tel: 34-91-708-08-90 Fax: 34-91-708-08-91 **Sweden - Gothenberg** Tel: 46-31-704-60-40 **Sweden - Stockholm** Tel: 46-8-5090-4654 **UK - Wokingham** Tel: 44-118-921-5800 Fax: 44-118-921-5820